

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Projekt dyplomowy

*Symulacyjny model adaptacyjnego tempomatu*

*A simulation model of adaptive cruise control system*

Autor: *Jakub Burczyk*

Kierunek studiów: Automatyka i Robotyka

Opiekun pracy: *dr inż., Marek Długosz*

Kraków, 2021

Spis treści

[1 Wstęp @TODO 3](#_Toc89128797)

[1.1 Charakterystyka problemu @TODO 3](#_Toc89128798)

[1.2 Cel oraz założenia pracy @TODO 3](#_Toc89128799)

[1.3 Zawartość pracy @TODO 3](#_Toc89128800)

[2 Środowisko symulacyjne – CARLA 3](#_Toc89128801)

[2.1 Czym jest CARLA 3](#_Toc89128802)

[2.2 Możliwości symulatora 3](#_Toc89128803)

[2.3 Interfejs symulatora 4](#_Toc89128804)

[3 Tworzenie środowisk testowych 5](#_Toc89128805)

[3.1 Standard OpenDRIVE 5](#_Toc89128806)

[3.2 Sposób zapisu pliku OpenDRIVE 5](#_Toc89128807)

[3.3 Oprogramowanie do edycji sieci OpenDRIVE 6](#_Toc89128808)

[3.3.1 MathWorks RoadRunner 6](#_Toc89128809)

[3.3.2 OddLOT 6](#_Toc89128810)

[4 Bibliografia 6](#_Toc89128811)

# Wstęp @TODO

## Charakterystyka problemu @TODO

@TODO

## Cel oraz założenia pracy @TODO

@TODO

## Zawartość pracy @TODO

@TODO

# Środowisko symulacyjne – CARLA

## Czym jest CARLA

Symulator CARLA (Car Learning to Act) jest środowiskiem symulacyjnym zaprojektowanym w celach badawczych i weryfikacyjnych systemów pojazdów autonomicznych. Jedym z celów przyświecających twórcom była całkowita transparentność dla użytkownika i mozliwość rozwoju projektu przez członków społeczności. Dostępny jest on w formie open-source pod licencją MIT[[1]](#footnote-1) zarówno na komputery z systemem Windows jak i Linux. Został zbudowany na bazie silnika Unreal Engine 4, który umożliwia wierną symulację oświetlenia i fizyki obiektów. W celu tworzenia otoczenia wewnątrz symulacji wykorzystywany jest standard OpenDRIVE, za pomocą którego definiowane są sieci dróg oraz ich parametry związane ze sterowaniem ruchem ulicznym.

## Możliwości symulatora

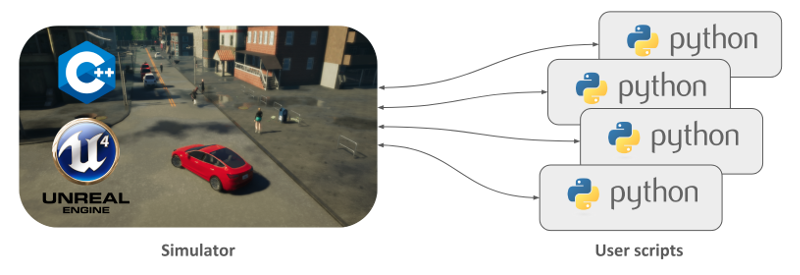
Dzięki zastosowaniu nowoczesnych rozwiązań, wsparciu gigantów technologicznych takich jak Intel, Samsung oraz producentów branży automotive takich jak Mercedes, Toyota czy Valeo symulator zaopatrzono w szereg zaawansowanych funkcjonalności.

Składają się na nie między innymi:

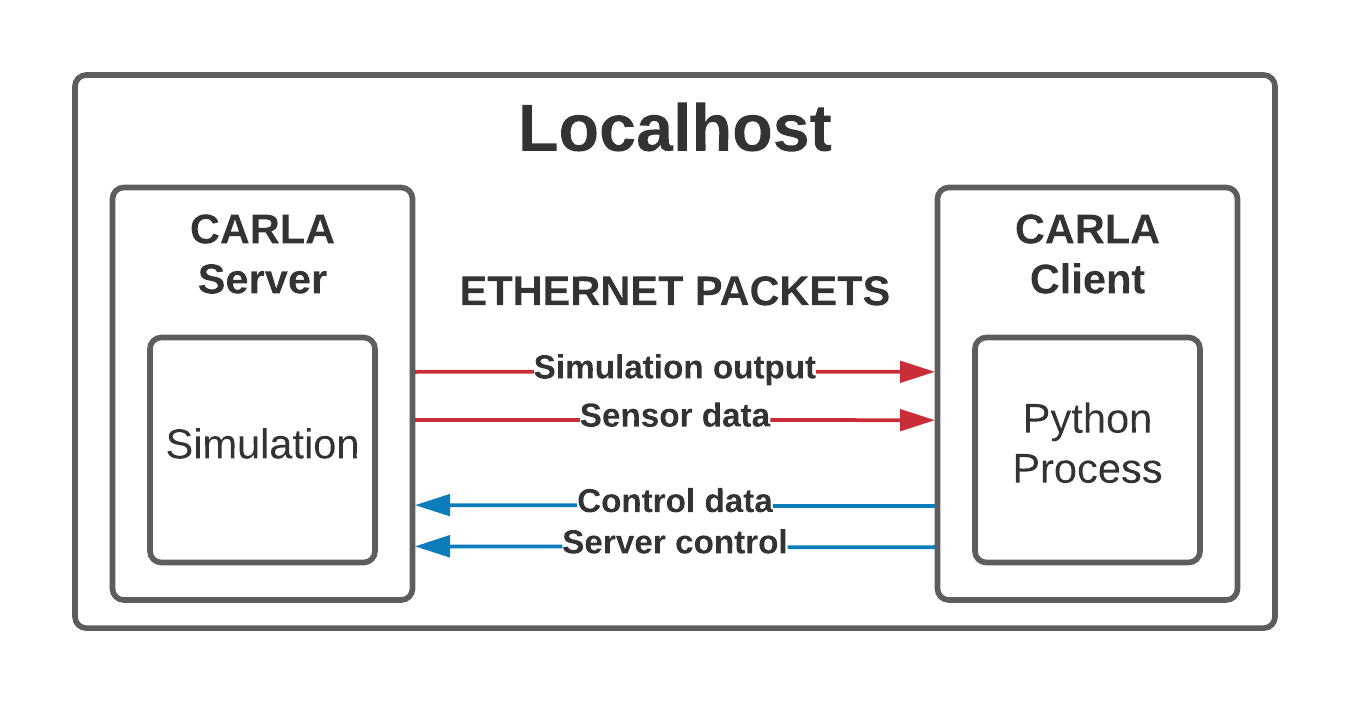
* Symulacja fizyki pojazdów, charakterystyk trakcji, oddziałujących sił i oporów
* Symulacja ruchu ulicznego przestrzegającego przepisów ruchu drogowego, znaków drogowych, sygnalizacji oraz reakcji na innych uczestników
* Symulacja sensorów i pseudosensorów np.:
  + Kamer
    - Głębi
    - RGB
    - Optical Flow – ruch pikseli pomiędzy klatkami
    - Segmentacja semantyczna – podział obrazu na klasy obiektów
  + GNSS – nawigacja satelitarna
  + IMU – jednostka pomiarowa zawierająca akcelerometr, żyroskop i kompas
  + Radar
  + LIDAR
  + Detektory:
    - Detektor kolizji
    - Detektor przekroczenia linii
    - Detektor przeszkód na drodze
* Połączenie z innymi systemami i srodowiskami np. opartymi o platformę ROS[[2]](#footnote-2)
* Tworzenie scenariuszy i środowisk testowych

## Interfejs symulatora

Symulator można podzielić na dwie części, serwerową i poszczególnych klientów. Część serwerowa odpowiedzialna jest między innymi za symulację sensorów, obliczenia fizyki obiektów, aktualizację świata i aktorów[[3]](#footnote-3). Klienci natomiast, za pośrednictwem API[[4]](#footnote-4) w języku Python, mogą komunikować się z serwerem poprzez wysyłanie komend i meta-komend zawierających informacje np. o sterowaniu lub zmianie parametrów symulacji. Obie części symulatora mogą działać równolegle na jednej maszynie, do której odwołujemy się poprzez hosta lokalnego, bądź poprzez sieć komputerową. Umożliwia to uruchomienie serwera na jednostce o dużej mocy obliczeniowej, która udostępnia klientom wysokiej jakości symulację, odciążając tym samym zasoby klientów, które mogą być przeznaczone na algorytmy sterowania.



Rysunek 2.1 Ogólny schemat architektury symulatora  
źródło: <https://carla.readthedocs.io/en/0.9.12/start_introduction/>



Rysunek 2.2 Schemat komunikacji w obrębie maszyny lokalnej

Należy zauważyć, że struktura ta może być rozszerzona o dowolną ilość klientów, z czego żaden nie jest limitowany co do ilości i rodzaju symulowanych obiektów.

# Tworzenie środowisk testowych

## Standard OpenDRIVE

Symulator CARLA domyślnie korzysta z otoczeń (map) zbudowanych z obiektów 3D. Pod tą warstwą wizualno–fizyczną znajdują się informacje o połączeniach, skrzyżowaniach, ilości i szerokości pasów ruchu, limitach prędkości itd., które przekazywane są do symulowanych pojazdów. Podporządkowanie się do tego standardu pozwala na przenoszenie informacji pomiędzy wieloma symulatorami. Został on zaadoptowany przez liderów branży takich jak BMW, dSPACE czy Vector Informatik.

## Sposób zapisu pliku OpenDRIVE

Dane zapisywane są jako plik tekstowy języka XML o rozszerzeniu .xodr. Każdy węzeł posiada informacje takie jak jego poprzednik, następca, typ obieku, pozycja i wymiary.

Obraz zawierający tekst

Opis wygenerowany automatycznie

Rysunek 3.1 Przykładowa droga i fragment pliku .xodr  
źródło: <https://www.asam.net/standards/detail/opendrive/>

Ze względu na charakterystykę tego opisu, bardzo szybko przestaje być on czytelny dla człowieka, dlatego do tworzenia sieci używa się wyspecjalizowanego oprogramowania.

## Oprogramowanie do edycji sieci OpenDRIVE

Duże możliwości opisu sieci dróg wymusiły powstanie oprogramowania do edycji wizualnej.

Powstało wiele, zarówno profesjonalnych, jak i amatorskich rozwiązań. Podstawowo każde z nich oferuje tworzenie i łączenie odcinków dróg oraz ustalanie skrzyżowań. Oprogramowanie profesjonalne zazwyczaj jest trudniejsze w obsłudze, ale udostępnia o wiele więcej funkcjonalności.

### MathWorks RoadRunner

RoadRunner jest prawdopodobnie najlepszym z dostępnych rozwiązań. Posiada bardzo rozbudowany edytor modeli 3D co pozwala na wierną reprezentację całego otoczenia wraz z mapami ukształtowania terenu, a nie jedynie informacji o drogach. Niestety jest jednym z płatnych edytorów, z tego względu nie został użyty w trakcie tworzenia projektu dyplomowego.

Obraz zawierający tekst, sprzęt elektroniczny

Opis wygenerowany automatycznie

Rysunek 3.2 Interfejs edytora RoadRunner  
źródło: [https://www.mathworks.com/products/roadrunner.html#road-and-3d](https://www.mathworks.com/products/roadrunner.html%23road-and-3d)

### OddLOT

# Bibliografia

1. A. Dosovitskiy, G. Ros, F. Codevilla, A. López, V. Koltun – „CARLA: An Open Urban Driving Simulator”; PMLR 78:1-16
2. Dokumentacja symulatora CARLA (w wersji 0.9.12) dostępna pod adresem: <https://carla.readthedocs.io/en/0.9.12/>

1. Licencja MIT – rodzaj licencji, który pozwala na dowolne modyfikacje i dystrybucję danego oprogramowania. [↑](#footnote-ref-1)
2. ROS – Robot Operating System – platforma programistyczna zaprojektowana z myślą o tworzeniu oprogramowania robotów źródło: <https://www.ros.org/> [↑](#footnote-ref-2)
3. Aktor – w rozumieniu symulatora CARLA, jest instancją obiektu symulacji, może być to między innymi pojazd, sensor, obserwator czy też obiekty związane ze sterowaniem ruchem drogowym [↑](#footnote-ref-3)
4. API – Application Programming Interface [↑](#footnote-ref-4)